What Is Claimed Is:

1. A method to facilitate code verification and garbage collection in a platform-independent virtual machine, comprising:

receiving a code module written in a platform-independent language;

examining the code module to locate a call to a program method within the code module; and

transforming the code module so that all operands remaining on an evaluation stack when the program method is called relate to the program method;

whereby verification and garbage collection of the code module is simplified.

2. The method of claim 1, wherein transforming the code module involves ensuring that local variables hold only values of a single type and do not hold variables of different types at different times.

3. The method of claim 1, wherein transforming the code module involves ensuring that the evaluation stack includes only elements related to a bytecode that may trigger garbage collection when the bytecode is executed.

4. The method of claim 1, wherein transforming the code module involves ensuring that only parameters for the program method are on the evaluation stack when the program method is called.

5. The method of claim 1, wherein transforming the code module further comprises spilling to memory stack slots that do not include operands for the call to the program method.

6. The method of claim 5, further comprising filling stack slots that were previously spilled upon return from the program method.

7. The method of claim 6, wherein the program method is associated with a single typemap to indicate a type for each variable on the evaluation stack.

8. An apparatus to facilitate code verification and garbage collection in a platform-independent virtual machine, comprising:

a receiving mechanism configured to receive a code module written in a platform-independent language;

an examining mechanism configured to examine the code module to locate a call to a program method within the code module; and

a transforming mechanism configured to transform the code module so that all operands remaining on an evaluation stack when the program method is called relate to the program method;

whereby verification and garbage collection of the code module is simplified.

9. The apparatus of claim 8, wherein transforming the code module involves ensuring that local variables hold only values of a single type and do not hold variables of different types at different times.

10. The apparatus of claim 8, wherein transforming the code module involves ensuring that the evaluation stack includes only elements related to a bytecode that may trigger garbage collection when the bytecode is executed.

11. The apparatus of claim 8, wherein transforming the code module involves ensuring that only parameters for the program method are on the evaluation stack when the program method is called.

12. The apparatus of claim 8, further comprising a spilling mechanism configured to spill to memory stack slots that do not include operands for the call to the program method when transforming the code module.

13. The apparatus of claim 12, further comprising a filling mechanism configured to fill stack slots that were previously spilled upon return from the program method.

14. The apparatus of claim 13, wherein the program method is associated with a single typemap to indicate a type for each variable on the evaluation stack.

15. A computer system to facilitate code verification and garbage collection in a platform-independent virtual machine, comprising:

a central processing unit;

a memory system;

a port for communicating with an external client;

a bus to couple the central processing unit, the memory system, and the port;

a receiving mechanism within the central processing unit configured to receive a code module written in a platform-independent language;

an examining mechanism configured to examine the code module to locate a call to a program method within the code module; and a transforming mechanism configured to transform the code module so that all operands remaining on an evaluation stack when the program method is called relate to the program method;

whereby verification and garbage collection of the code module is simplified.

16. The computer system of claim 15, wherein transforming the code module involves ensuring that local variables hold only values of a single type and do not hold variables of different types at different times.

17. The computer system of claim 15, wherein transforming the code module involves ensuring that the evaluation stack includes only elements related to a bytecode that may trigger garbage collection when the bytecode is executed.

18. The computer system of claim 15, wherein transforming the code module involves ensuring that only parameters for the program method are on the evaluation stack when the program method is called.

19. The computer system of claim 15, further comprising a spilling mechanism configured to spill to memory stack slots that do not include operands for the call to the program method when transforming the code module.

20. The computer system of claim 19, further comprising a filling mechanism configured to fill stack slots that were previously spilled upon return from the program method.

21. The computer system of claim 20, wherein the program method is associated with a single typemap to indicate a type for each variable on the evaluation stack.